# Correlation

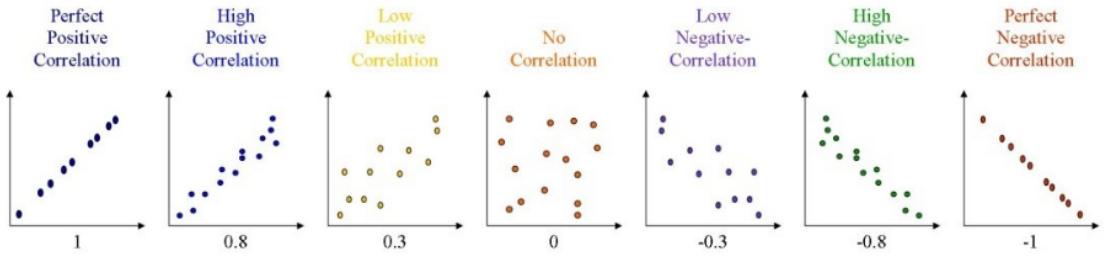
November 27, 2024

## 0.1 Correlation

Correlation answers questions such as how one variable changes with respect to another. If it does change, then to what degree or strength? Additionally, if the relation between those variables is strong enough, then we can make predictions for future behavior. Correlation tells us how variables change together, both in the same or opposite directions and in the magnitude (that is, strength) of the relationship. To find the correlation, we calculate the Pearson correlation coefficient, symbolized by  (the Greek letter rho). This is obtained by dividing the covariance by the product of the standard deviations of the variables:

$$\rho xy = \frac{\sigma xy}{\sigma x \sigma y}$$

In terms of the strength of the relationship, the value of the correlation between two variables, A and B, varies between +1 and -1. If the correlation is +1, then it is said to be a perfect positive/linear correlation (that is, variable A is directly proportional to variable B), while a correlation of -1 is a perfect negative correlation (that is, variable A is inversely proportional to variable B). Note that values closer to 0 are not supposed to be correlated at all. If correlation coefficients are near to 1 in absolute value, then the variables are said to be strongly correlated; in comparison, those that are closer to 0.5 are said to be weakly correlated.




Here, are the key observations:

1. When the data points plot has a straight line going through the origin to the x and y values, then the variables are said to have a positive correlation.

2. When the data points plot to generate a line that goes from a high value on the yaxis to a high value on the x axis, the variables are said to have a negative correlation.
3. A perfect correlation has a value of 1.
4. A perfect negative correlation has a value of -1.

A highly positive correlation is given a value closer to 1. A highly negative correlation is given a value closer to -1. In the preceding diagram, +0.8 gives a high positive correlation and -0.8 gives a high negative correlation. The closer the number is to 0 (in the diagram, this is +0.3 and -0.3), the weaker the correlation.

# 1 Types of Analysis

1. Univariate Analysis
2. Bivariate Analysis
3. Multivariate Analysis

[ ]:
```
#import libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

[ ]:
```
plt.rcParams['figure.figsize'] = (6, 4)
plt.rcParams['figure.dpi'] = 150
```

[3]:
```
# loading data set as Pandas dataframe
df = pd.read_csv("https://raw.githubusercontent.com/PacktPublishing/
 ↪hands-on-exploratory-data-analysis-with-python/master/Chapter%207/automobile.
 ↪csv")
df.head()
```

[3]:

| | symboling | normalized-losses | make | … | city-mpg | highway-mpg | price |
|---|---|---|---|---|---|---|---|
| 0 | 3 | ? | alfa-romero | … | 21 | 27 | 13495 |
| 1 | 3 | ? | alfa-romero | … | 21 | 27 | 16500 |
| 2 | 1 | ? | alfa-romero | … | 19 | 26 | 16500 |
| 3 | 2 | 164 | audi | … | 24 | 30 | 13950 |
| 4 | 2 | 164 | audi | … | 18 | 22 | 17450 |

[5 rows x 26 columns]

[4]:
```
df.dtypes
```

[4]:
```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
```

```
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

## 2 Data Cleaning

[5]:
```
# Find out the number of values which are not numeric
df['price'].str.isnumeric().value_counts()

# List out the values which are not numeric
df['price'].loc[df['price'].str.isnumeric() == False]

#Setting the missing value to mean of price and convert the datatype to integer
price = df['price'].loc[df['price'] != '?']
pmean = price.astype(int).mean()
df['price'] = df['price'].replace('?',pmean).astype(int)
df['price'].head()
```

[5]:
```
0    13495
1    16500
2    16500
3    13950
4    17450
Name: price, dtype: int64
```

[6]:
```
# Cleaning the horsepower losses field
df['horsepower'].str.isnumeric().value_counts()
horsepower = df['horsepower'].loc[df['horsepower'] != '?']
```

```
hpmean = horsepower.astype(int).mean()
df['horsepower'] = df['horsepower'].replace('?',hpmean).astype(int)
df['horsepower'].head()
```

[6]:
```
0    111
1    111
2    154
3    102
4    115
Name: horsepower, dtype: int64
```

[7]:
```
# Cleaning the Normalized losses field
df[df['normalized-losses']=='?'].count()
nl=df['normalized-losses'].loc[df['normalized-losses'] !='?'].count()
nmean=nl.astype(int).mean()
df['normalized-losses'] = df['normalized-losses'].replace('?',nmean).astype(int)
df['normalized-losses'].head()
```

[7]:
```
0    164
1    164
2    164
3    164
4    164
Name: normalized-losses, dtype: int64
```

Now computing the Measure of central tendency of the values in column height. Remember taking only a single column of the data set we are making a univariate analysis.

[8]:
```
#calculate mean, median and mode of dat set height
mean = df["height"].mean()
median =df["height"].median()
mode = df["height"].mode()
print(mean , median, mode)
```

```
53.724878048780525 54.1 0    50.8
dtype: float64
```

### 2.0.1 Univariate Analysis

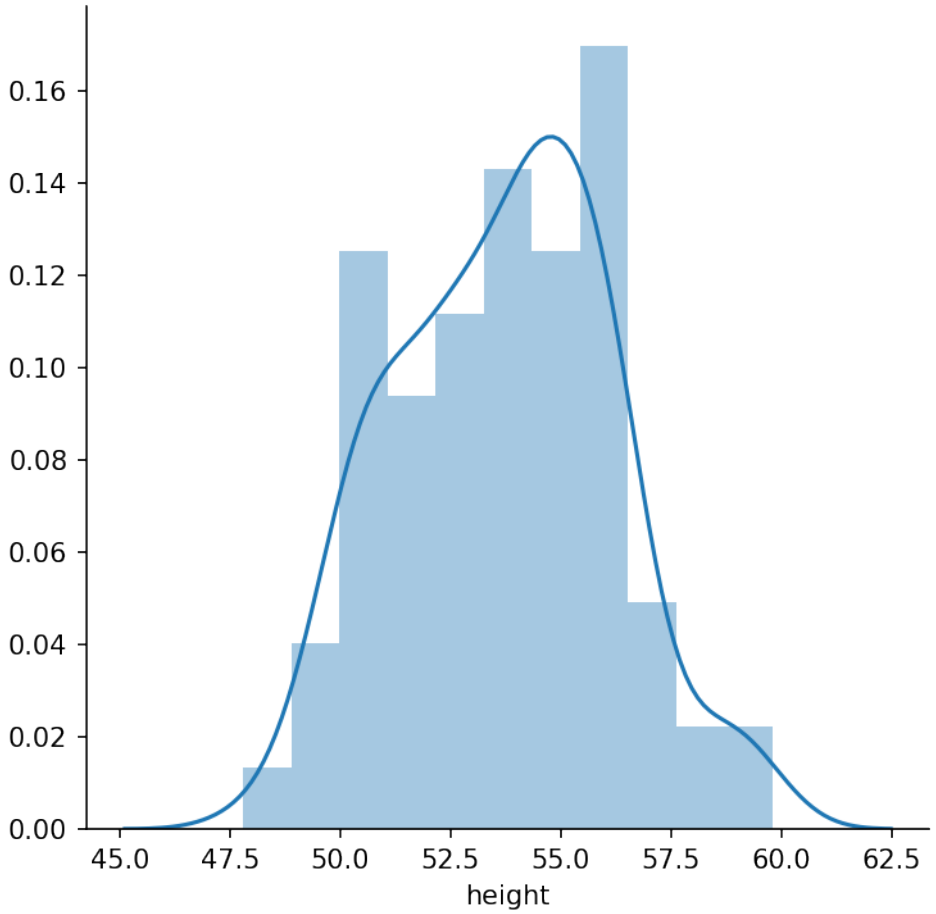
Analysis on a single type of dataset is called univariate analysis. Univariate analysis is the simplest form of analyzing data. It means that our data has only one type of variable and that we perform analysis over it. The main purpose of univariate analysis is to take data, summarize that data, and find patterns among the values. It doesn't deal with causes or relationships between the values. Several techniques that describe the patterns found in univariate data include central tendency (that is the mean, mode, and median) and dispersion (that is, the range, variance, maximum and minimum quartiles (including the interquartile range), and standard deviation)

Now let's visualize this analysis in graph.

[9]:
```
#distribution plot
sns.FacetGrid(df,size=5).map(sns.distplot,"height").add_legend()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:243: UserWarning: The
`size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

[9]:
```
<seaborn.axisgrid.FacetGrid at 0x7f6ab7e15be0>
```




From the above graph, we can observe that the hight of maximum cars ranges from 53 to 57.

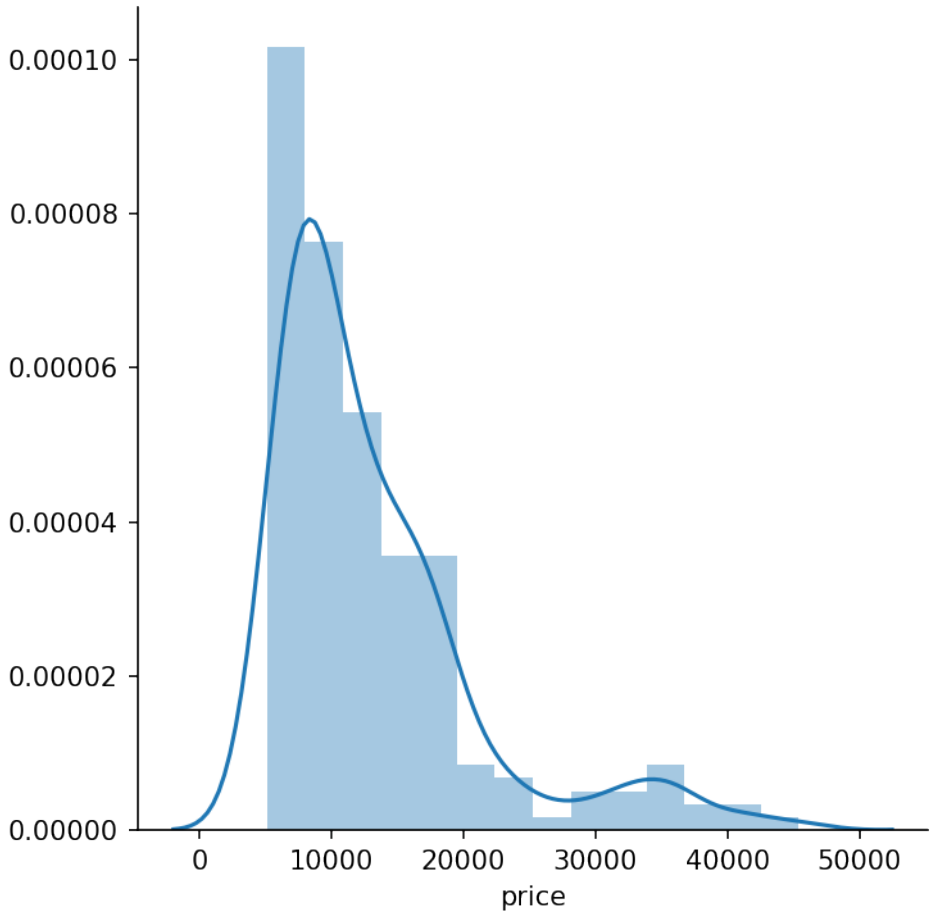
Now, let's draw the same with price. Since data type of price is object so first change object to numeric data type.

[10]:
```
#distribution plot
sns.FacetGrid(df,size=5).map(sns.distplot,"price").add_legend()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:243: UserWarning: The
`size` parameter has been renamed to `height`; please update your code.
```

```
  warnings.warn(msg, UserWarning)
```

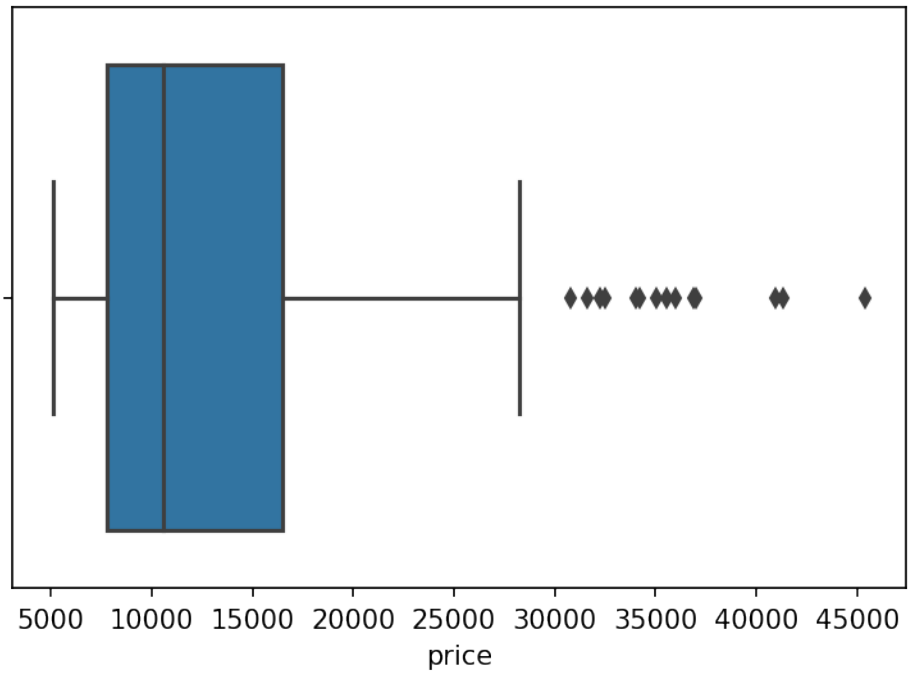
[10]: `<seaborn.axisgrid.FacetGrid at 0x7f6ab54d73c8>`




From the above graph, we can say that the price ranges from 5,000 to 45,000 but the maximum car price ranges between 5,000 to 10,000.

The box plot is also efective visual representation of statical measures like median and quartiles in univariate analysis.

[11]:
```
#boxplot for price of cars
sns.boxplot(x="price",data=df)
plt.show()
```



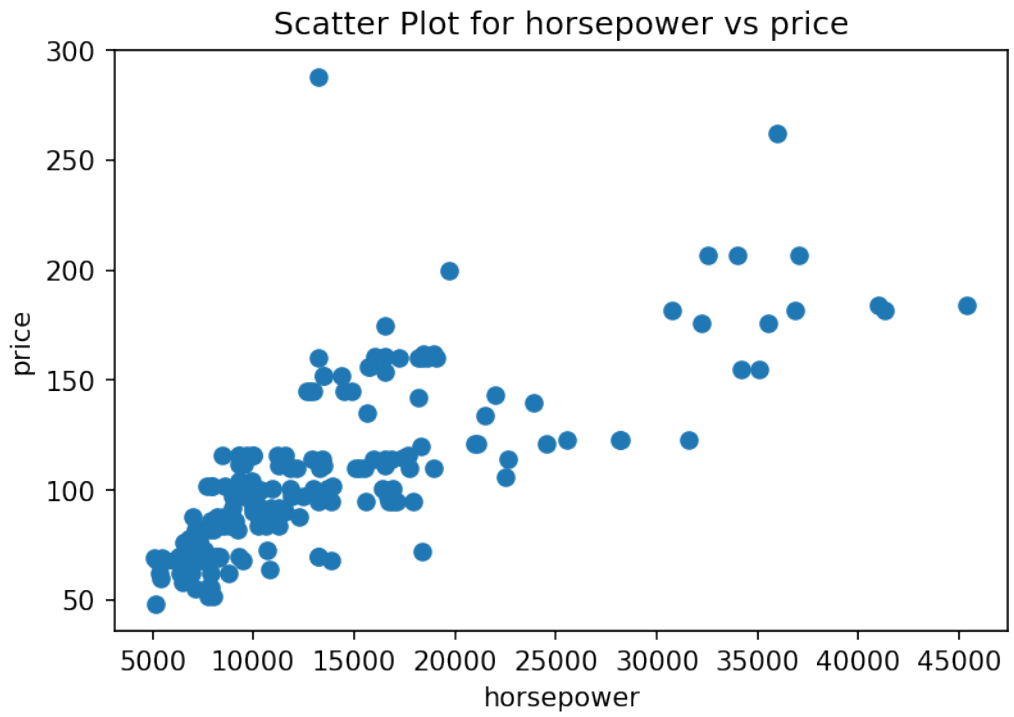

The right border of the box is Q3 and the left border of the box is Q1. Lines extend from both sides of the box boundaries toward the minimum and maximum. Based on the convention our plotting tool uses, though, they may only extend to a certain statistic; any values beyond these statistics are marked as outliers (using points).[Referance:HANDSON_DATA_ANALYSIS_WITH_PANDAS.pdf]

### 2.0.2 Bivariate Analysis

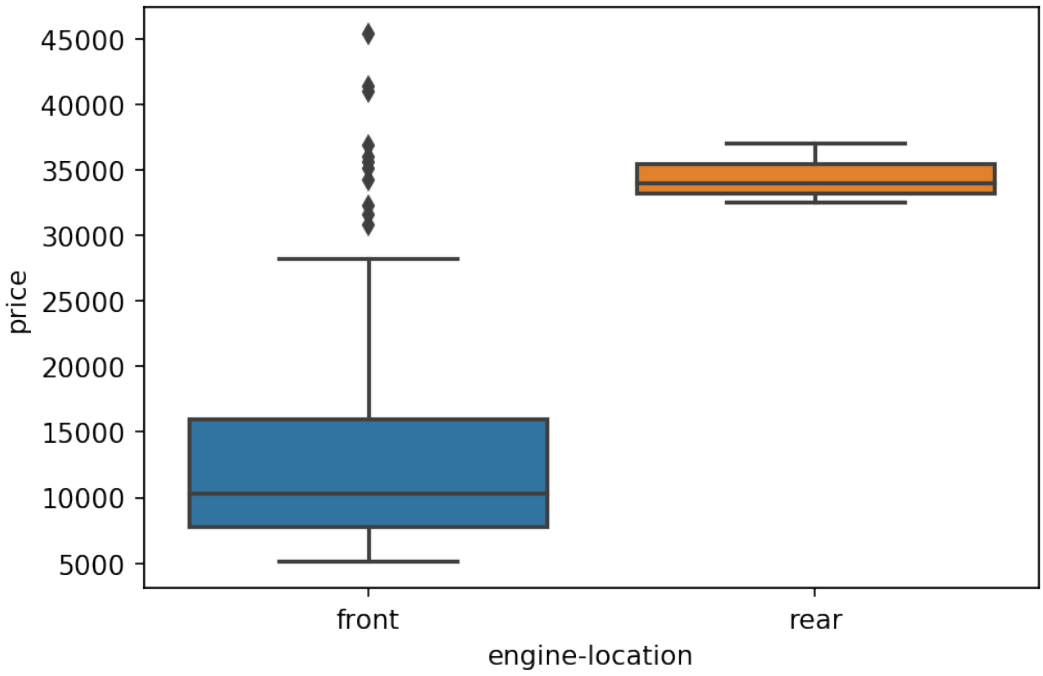
As its name suggests, this is the analysis of more than one (that is, exactly two) type of variable. Bivariate analysis is used to find out whether there is a relationship between two different variables. When we create a scatter plot by plotting one variable against another on a Cartesian plane (think of the x and y axes), it gives us a picture of what the data is trying to tell us. If the data points seem to fit the line or curve, then there is a relationship or correlation between the two variables. Generally, bivariate analysis helps us to predict a value for one variable (that is, a dependent variable) if we are aware of the value of the independent variable.

[12]:
```
# plot the relationship between "horsepower" and "price"
plt.scatter(df["price"], df["horsepower"])
plt.title("Scatter Plot for horsepower vs price")
plt.xlabel("horsepower")
plt.ylabel("price")
```

[12]:
```
Text(0, 0.5, 'price')
```




[13]:
```
#boxplot
sns.boxplot(x="engine-location",y="price",data=df)
plt.show()
```



[14]:
```
#boxplot to visualize the distribution of "price" with types of "drive-wheels"
sns.boxplot(x="drive-wheels", y="price",data=df)
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ab4e827b8>
```

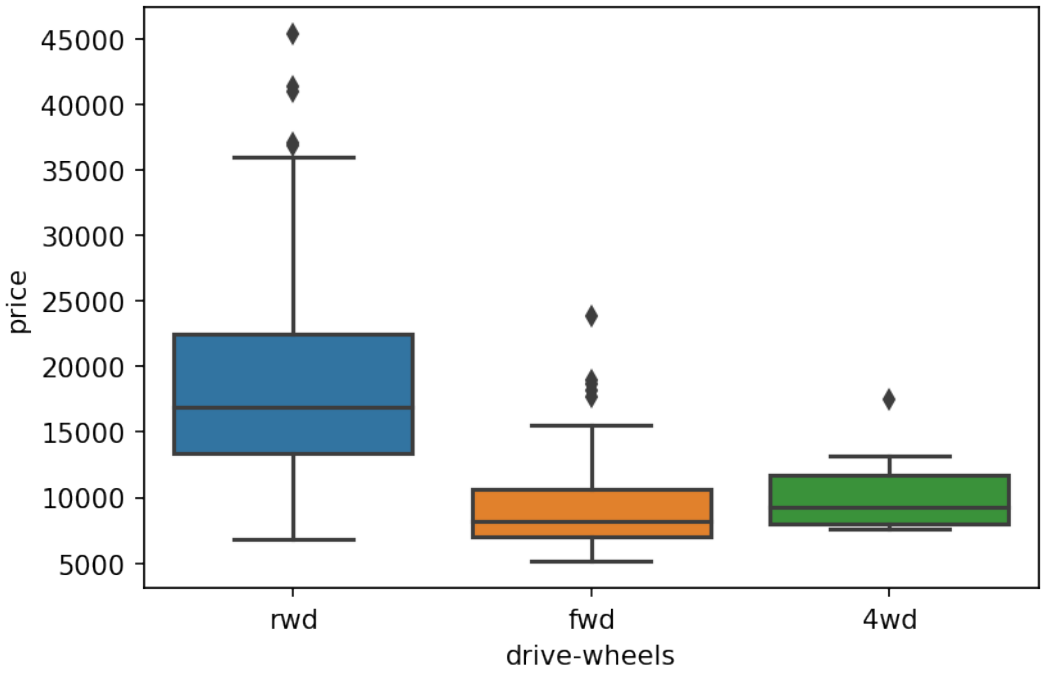
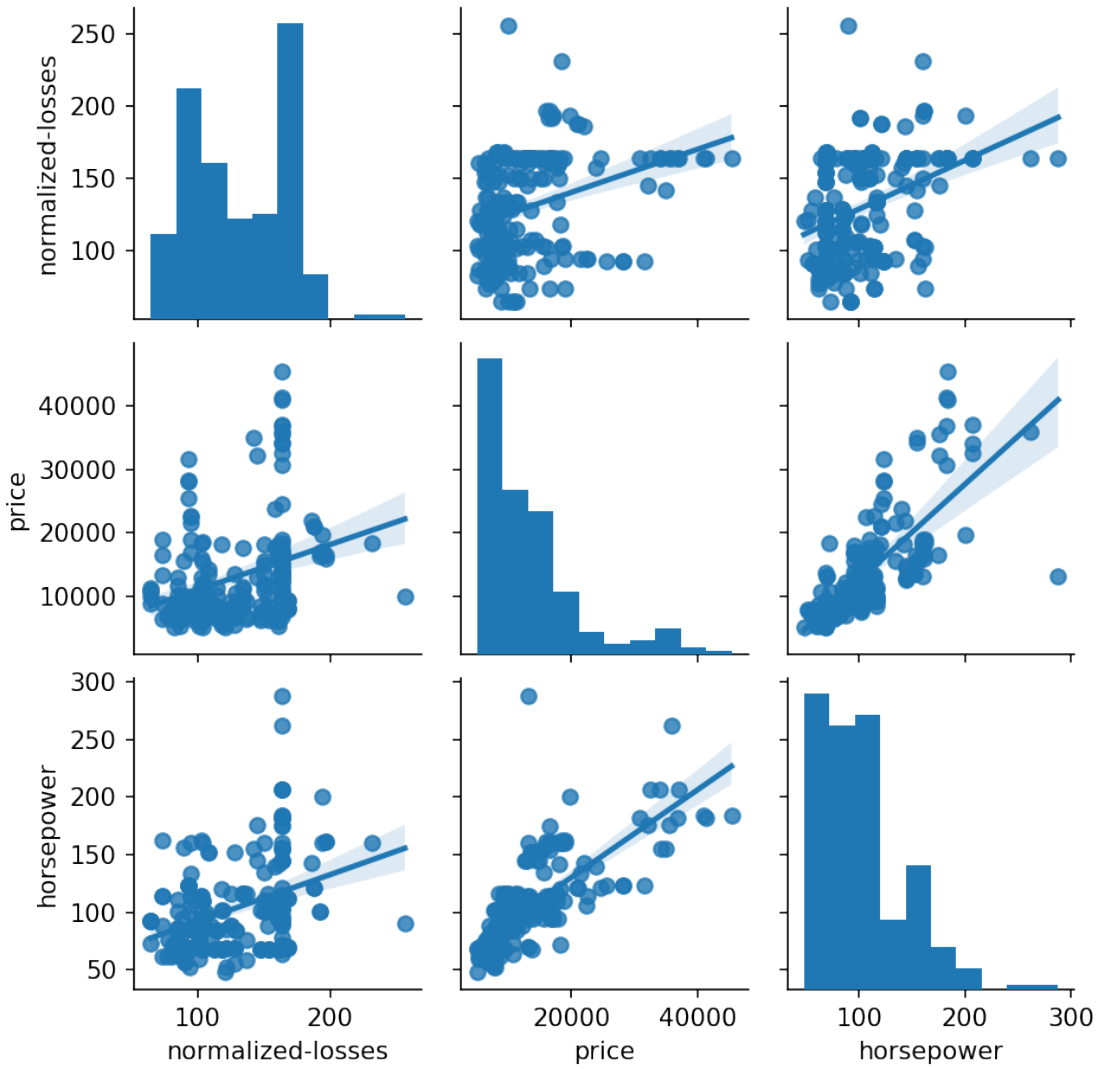



Figure above illustrates the range of prices in cars with different wheel types. Boxplot shows the average and median price in respective wheel types and some outliers.

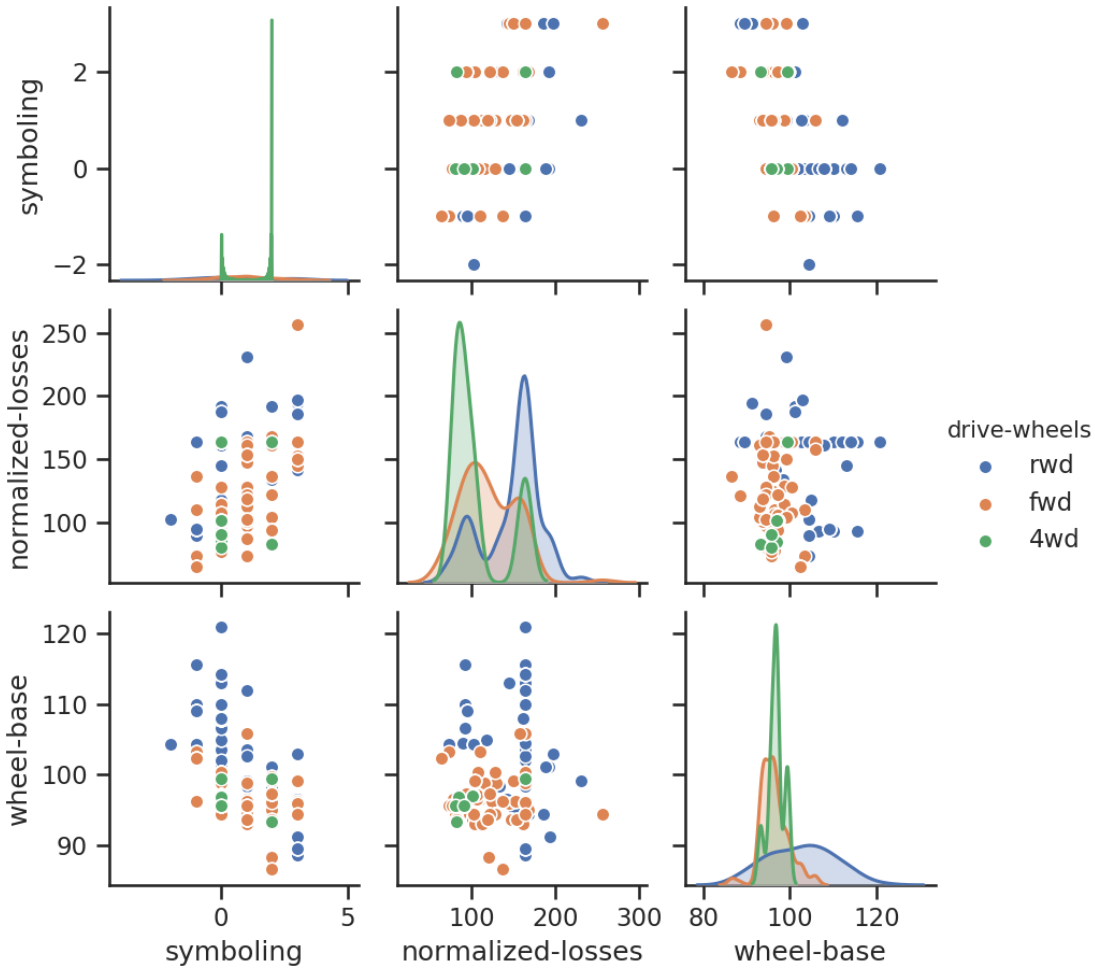
### 2.0.3 Multivariate Analysis

Multivariate analysis is the analysis of three or more variables. This allows us to look at correlations (that is, how one variable changes with respect to another) and attempt to make predictions for future behavior more accurately than with bivariate analysis.

[15]:
```
# pair plot with plot type regression
sns.pairplot(df,height=2, vars = ['normalized-losses', 'price','horsepower'], ␣
 ↪kind="reg")
plt.show()
```




[16]:
```
 #pair plot (matrix scatterplot) of few columns
sns.set(style="ticks", color_codes=True)
sns.pairplot(df,height=2,vars = ['symboling',␣
 ↪'normalized-losses','wheel-base'], hue="drive-wheels")
plt.show()
```




[17]:
```
from scipy import stats

corr = stats.pearsonr(df["price"], df["horsepower"])
print("p-value:\t", corr[1])
print("cor:\t\t", corr[0])
```

```
p-value:        1.5910332446597316e-39
cor:            0.757945621793524
```

Here the correlation of these two variable is 0.80957 which is close to +1 thus we can make sure that price and horsepower are highly positively correlated. Using pandas corr( function correlation between entire numerical record can be calculated.

[18]:
```
correlation = df.corr(method='pearson')
correlation
```

[18]:

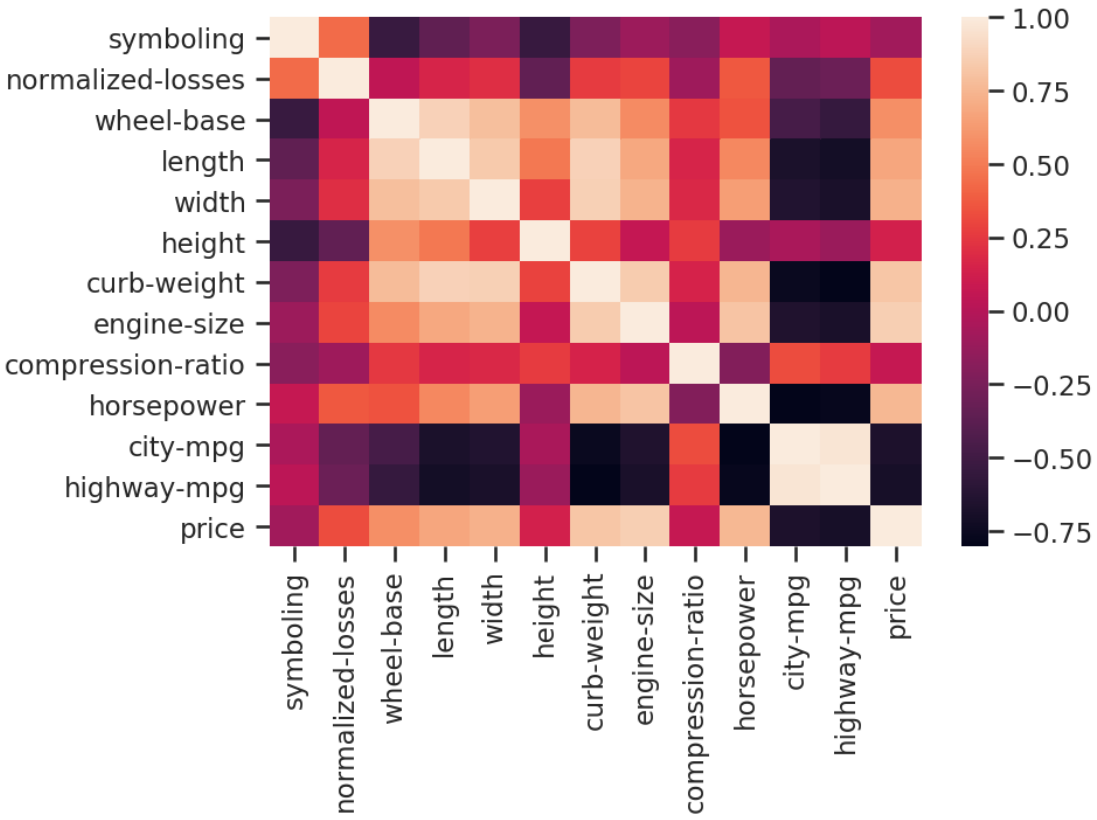
| | symboling | normalized-losses | … | highway-mpg | price |
|---|---|---|---|---|---|
| symboling | 1.000000 | 0.442093 | … | 0.034606 | -0.082201 |
| normalized-losses | 0.442093 | 1.000000 | … | -0.307189 | 0.326489 |
| wheel-base | -0.531954 | 0.042699 | … | -0.544082 | 0.583168 |
| length | -0.357612 | 0.155090 | … | -0.704662 | 0.682986 |
| width | -0.232919 | 0.209908 | … | -0.677218 | 0.728699 |
| height | -0.541038 | -0.346399 | … | -0.107358 | 0.134388 |
| curb-weight | -0.227691 | 0.262187 | … | -0.797465 | 0.820825 |
| engine-size | -0.105790 | 0.300268 | … | -0.677470 | 0.861752 |
| compression-ratio | -0.178515 | -0.097432 | … | 0.265201 | 0.070990 |
| horsepower | 0.071380 | 0.371238 | … | -0.770905 | 0.757946 |
| city-mpg | -0.035823 | -0.344018 | … | 0.971337 | -0.667449 |
| highway-mpg | 0.034606 | -0.307189 | … | 1.000000 | -0.690526 |
| price | -0.082201 | 0.326489 | … | -0.690526 | 1.000000 |

[13 rows x 13 columns]

Now let's visualize this correlation analysis with heatmap. Heatmap is best technique to make this look beautiful and easier to interpret.

[19]:
```
sns.heatmap(correlation,xticklabels=correlation.columns,
            yticklabels=correlation.columns)
```

[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ab2f37f98>
```




A coefficient close to 1 means that there's a very strong positive correlation between the two variables. The diagonal line is the correlation of the variables to themselves — so they'll obviously be 1.

**Multivariate Analysis over titanic dataset**

[20]:
```
#load data set
titanic=pd.read_csv("https://raw.githubusercontent.com/PacktPublishing/
 ↪hands-on-exploratory-data-analysis-with-python/master/Chapter%207/titanic.
 ↪csv")
titanic.head()
```

[20]:

| | PassengerId | Survived | Pclass | … | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | … | 7.2500 | NaN | S |
| 1 | 2 | 1 | 1 | … | 71.2833 | C85 | C |
| 2 | 3 | 1 | 3 | … | 7.9250 | NaN | S |
| 3 | 4 | 1 | 1 | … | 53.1000 | C123 | S |
| 4 | 5 | 0 | 3 | … | 8.0500 | NaN | S |

[5 rows x 12 columns]

[21]:
```
titanic.shape
```

[21]:
```
(891, 12)
```

Let's take a look at what is the number of records missing in the data set.

[22]:
```
total = titanic.isnull().sum().sort_values(ascending=False)
total
```

[22]:
```
Cabin          687
Age            177
Embarked         2
Fare             0
Ticket           0
Parch            0
SibSp            0
Sex              0
Name             0
Pclass           0
Survived         0
PassengerId      0
dtype: int64
```

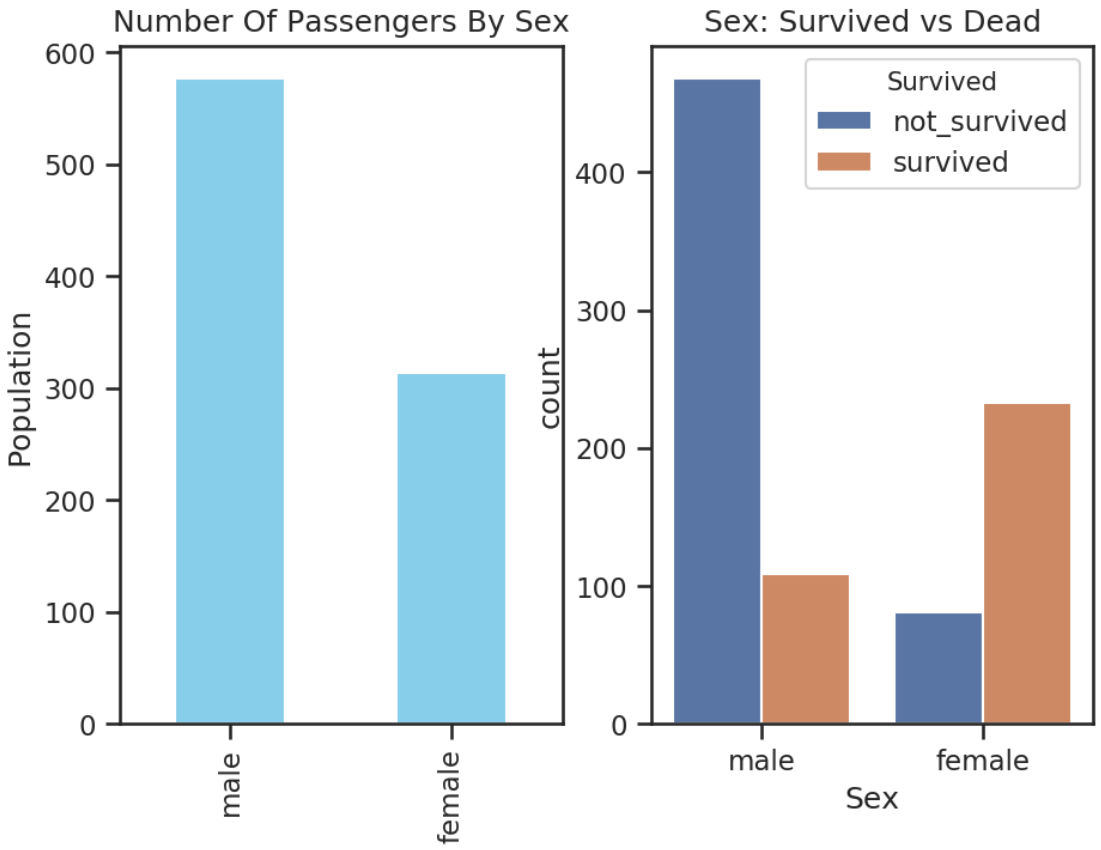
[23]:
```
#percentage of women survived
women = titanic.loc[titanic.Sex == 'female']["Survived"]
rate_women = sum(women)/len(women)

#percentage of men survived
men = titanic.loc[titanic.Sex == 'male']["Survived"]
rate_men = sum(men)/len(men)

print(str(rate_women) +" % of women who survived." )
print(str(rate_men) + " % of men who survived." )
```

```
0.7420382165605095 % of women who survived.
0.18890814558058924 % of men who survived.
```

You can see the number of females survival was high, so gender could be the attribute that contributes to analyzing the survival of any variable(person). Let's visualize this information on survival numbers in males and females.

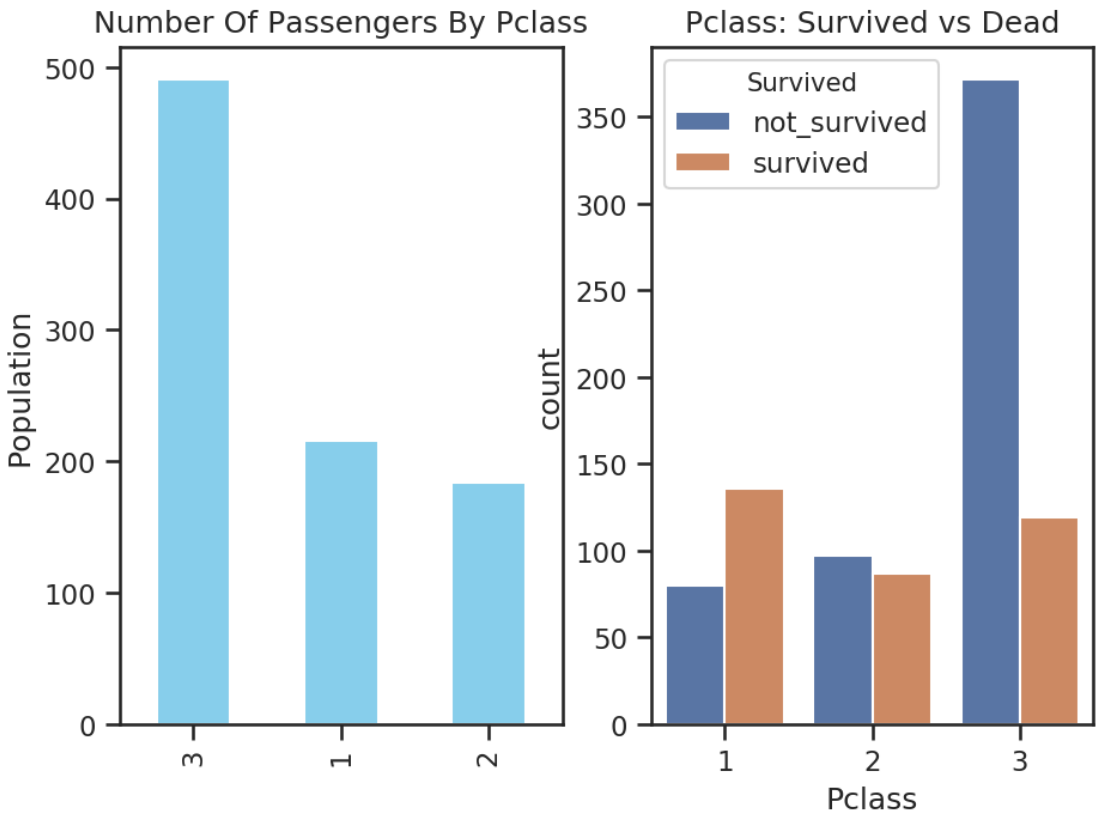
[24]:
```
titanic['Survived'] = titanic['Survived'].map({0:"not_survived", 1:"survived"})

fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Sex"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Sex")
ax[0].set_ylabel("Population")
sns.countplot("Sex", hue = "Survived", data = titanic, ax = ax[1])
```

```
ax[1].set_title("Sex: Survived vs Dead")
plt.show()
```




Let's visualize the number of survival and death from different Pclasses.

[25]:
```
fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Pclass"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Pclass")
ax[0].set_ylabel("Population")
sns.countplot("Pclass", hue = "Survived", data = titanic, ax = ax[1])
ax[1].set_title("Pclass: Survived vs Dead")
plt.show()
```




Looks like the number of passenger in Pclass 3 was high and maximum of them could not survive. death Pclass the number of death is high. And in Pclass 1 maximum of the passengers were survived.

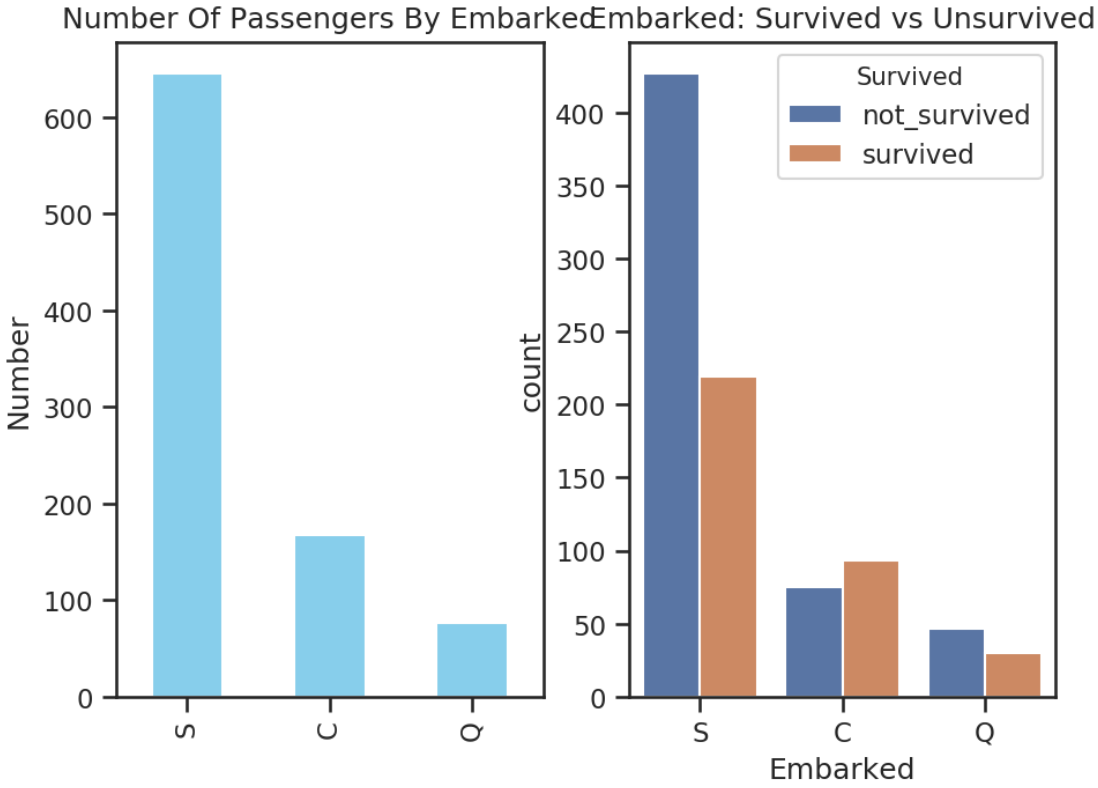
[26]:
```
titanic["Embarked"] = titanic["Embarked"].fillna("S")
titanic
```

[26]:

| | PassengerId | Survived | Pclass | … | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|
| 0 | 1 | not_survived | 3 | … | 7.2500 | NaN | S |
| 1 | 2 | survived | 1 | … | 71.2833 | C85 | C |
| 2 | 3 | survived | 3 | … | 7.9250 | NaN | S |
| 3 | 4 | survived | 1 | … | 53.1000 | C123 | S |
| 4 | 5 | not_survived | 3 | … | 8.0500 | NaN | S |
| .. | … | … | … | … | … | … | … |
| 886 | 887 | not_survived | 2 | … | 13.0000 | NaN | S |
| 887 | 888 | survived | 1 | … | 30.0000 | B42 | S |
| 888 | 889 | not_survived | 3 | … | 23.4500 | NaN | S |
| 889 | 890 | survived | 1 | … | 30.0000 | C148 | C |
| 890 | 891 | not_survived | 3 | … | 7.7500 | NaN | Q |

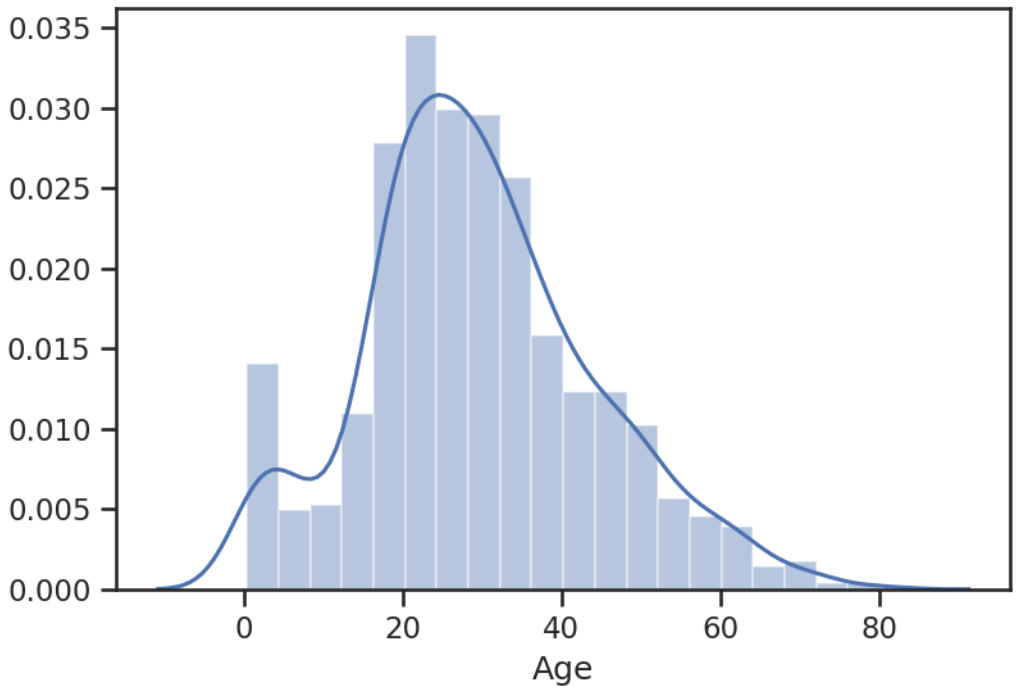
[891 rows x 12 columns]

[27]:
```
fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Embarked"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Embarked")
ax[0].set_ylabel("Number")
sns.countplot("Embarked", hue = "Survived", data = titanic, ax = ax[1])
ax[1].set_title("Embarked: Survived vs Unsurvived")
plt.show()
```



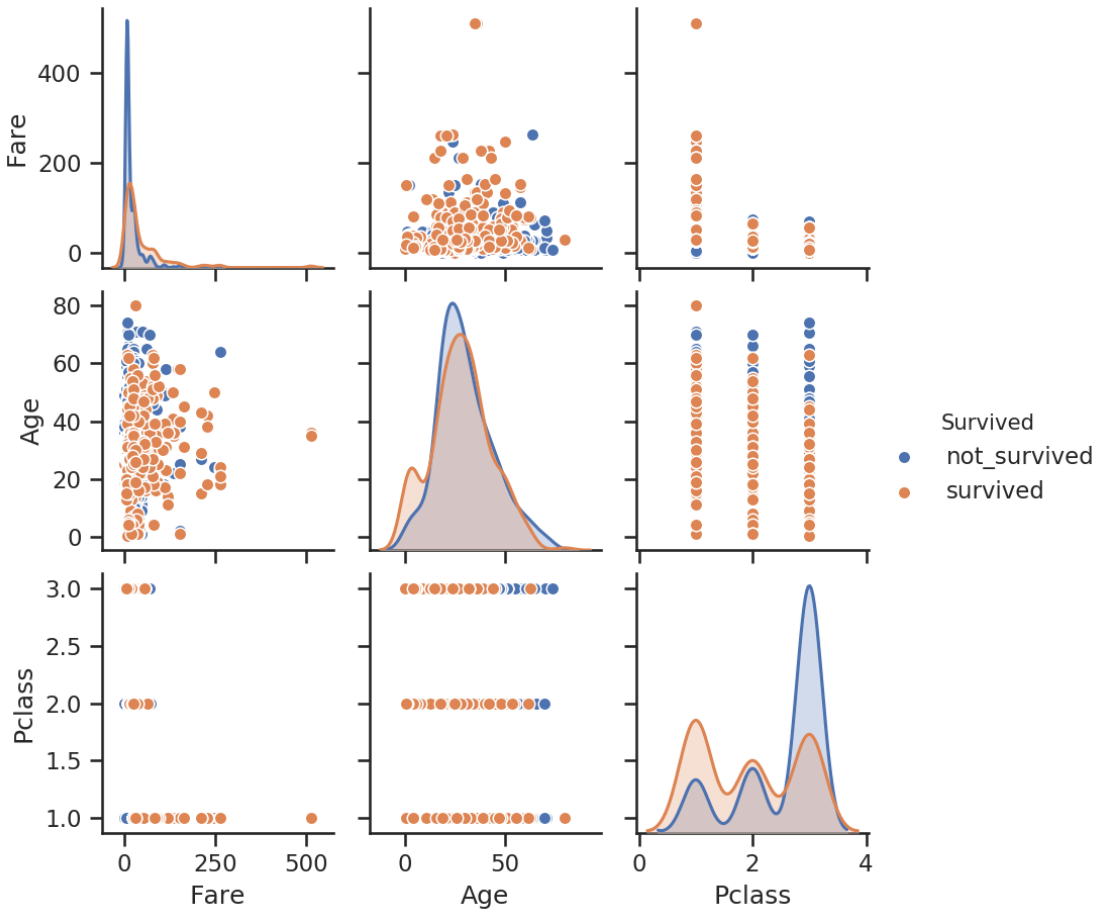

[28]:
```
sns.distplot(titanic['Age'].dropna())
```

[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ab1925860>
```




Now let's do first multivariate analysis into titanic data set with variables Survived, Pclass,Fear and Age.

[29]:
```
sns.set(style="ticks", color_codes=True)
sns.pairplot(titanic,height=2,vars = [ 'Fare','Age','Pclass'], hue="Survived")
plt.show()
```




Now lets' view the correlation table with heatmap. But first map Embarked records with integer values so that we can include Embrake too in our correlation analysis.

[30]:
```
titanic['Embarked'] = titanic['Embarked'].map({"S":1, "C":2,"Q":2,"NaN":0})
Tcorrelation = titanic.corr(method='pearson')
Tcorrelation
```

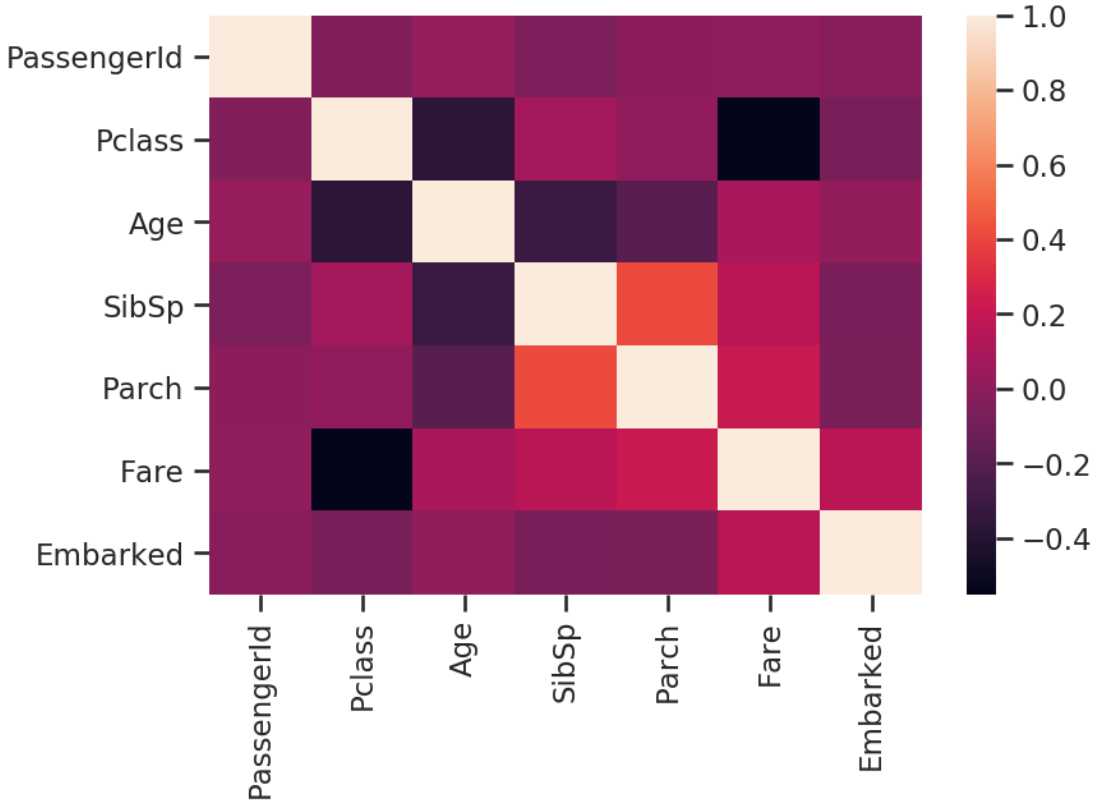
[30]:

| | PassengerId | Pclass | Age | … | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|
| PassengerId | 1.000000 | -0.035144 | 0.036847 | … | -0.001652 | 0.012658 | -0.022204 |
| Pclass | -0.035144 | 1.000000 | -0.369226 | … | 0.018443 | -0.549500 | -0.074053 |
| Age | 0.036847 | -0.369226 | 1.000000 | … | -0.189119 | 0.096067 | 0.023233 |
| SibSp | -0.057527 | 0.083081 | -0.308247 | … | 0.414838 | 0.159651 | -0.068734 |
| Parch | -0.001652 | 0.018443 | -0.189119 | … | 1.000000 | 0.216225 | -0.060814 |
| Fare | 0.012658 | -0.549500 | 0.096067 | … | 0.216225 | 1.000000 | 0.162184 |
| Embarked | -0.022204 | -0.074053 | 0.023233 | … | -0.060814 | 0.162184 | 1.000000 |

[7 rows x 7 columns]

[31]:
```
sns.heatmap(Tcorrelation,xticklabels=Tcorrelation.columns,
            yticklabels=Tcorrelation.columns)
```

[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6ab1a66320>




## 2.1 Simpson's paradox

Simpson's paradox is the difference that appears in a trend of analysis when a dataset is analyzed in two different situations: first, when data is separated into groups and, second, when data is aggregated.